MACHINE PROBLEM 1

1. Consider the example on the experiment of coin tossing. A fair coin is tossed 100 times with the relative frequency of heads tabulated as $S_0, S_1, \cdots, S_{100}$ based on the number of heads. Simulate 1,000 repetitions of the experiment and obtain the data on the number of trials $N_{1000}(S_k)$ where S_k is the number of heads. For example $N_{1000}(S_{50}) = 75$, means 75 trials yielded 50 heads.

(a) Obtain the following data on the sum of the no. of trials for each group (of 3) with the specified no. of heads S_k :

$N_{1000}(S_{33}) + N_{1000}(S_{34}) + N_{1000}(S_{35}) =$; $N_{1000}(S_{51}) + N_{1000}(S_{52}) + N_{1000}(S_{53}) =$;

$N_{1000}(S_{36}) + N_{1000}(S_{37}) + N_{1000}(S_{38}) =$; $N_{1000}(S_{54}) + N_{1000}(S_{55}) + N_{1000}(S_{56}) =$;

$N_{1000}(S_{39}) + N_{1000}(S_{40}) + N_{1000}(S_{41}) =$; $N_{1000}(S_{57}) + N_{1000}(S_{58}) + N_{1000}(S_{59}) =$;

$N_{1000}(S_{42}) + N_{1000}(S_{43}) + N_{1000}(S_{44}) =$; $N_{1000}(S_{60}) + N_{1000}(S_{61}) + N_{1000}(S_{62}) =$;

$N_{1000}(S_{45}) + N_{1000}(S_{46}) + N_{1000}(S_{47}) =$; $N_{1000}(S_{63}) + N_{1000}(S_{64}) + N_{1000}(S_{65}) =$;

$N_{1000}(S_{48}) + N_{1000}(S_{49}) + N_{1000}(S_{50}) =$; $N_{1000}(S_{66}) + N_{1000}(S_{67}) + N_{1000}(S_{68}) =$;

(b) Using the data in (a) by using a bar graph, plot the average number of heads per group vs. k, where k is the number of trials per group. Does the graph resemble a bell curve? Why and why not?

(c) Using the results of the experiment, compute the probability, expected value and standard deviation of X - the no. of heads per trial (100 tosses). Perform 20 runs and print the results.

Hint: For faster generation of random 0's and 1's, use the randint feature in Numpy:

Example:

```
In [96]: np.random.randint(2,size=(3,10))
Out[96]:
array([[0, 1, 0, 1, 1, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 1, 0, 0, 0, 1],
       [1, 0, 1, 1, 0, 1, 1, 0, 1, 1]])
```

2. A particular system in operation has 100 components. Each component has a failure probability q independent of the other components. The system is operational if any one of conditions a and b is satisfied:

a) Components $1, 2, \cdots, 40$ all work, and, any one of components 41, 42, ⋯, 50 works.

b) All of the components $51, 52, \cdots, 80$ works and any one of components 81, 82, ⋯, 100 works.

2.1) What is the probability that the system is operational? Write function system_operational (...) in Python 3 code that computes for the system probability.

2.2) Suppose we can replace any 20 of these components with an equivalent no. of ultra-reliable components, each of which has a failure probability of $q/10$. Which components must be prioritized in the replacement to increase system reliability?

2.3) Suppose the failure probability of components in (a) is better than the components in (b). That is, $q_a = 0.67q_b$, where $q_a$and q_b are the failure probabilities of each of the components in (a) and (b), respectively. Determine which components should be replaced with the ultra-reliable component $(0.20q_a)$ at minimum cost.

Questions 2.2 and 2.3 are independent of each other. Solve 2.1, 2.2 and 2.3 using simulation with $q = q_a = 0.275$. For each replacement of a regular component, is it necessary to perform 100 trials? Are 100 trials sufficient to conclude which component should be replaced?

3. Recall the case study example in lecture 3, the Pratt-Woodruff experiment where some 32 students were made to guess five cards with different ESP symbols. The subject picks up one card, concentrates, and guesses the symbol on the card. Out of 60,000 guesses, 12,489 were correct. The probability of this happening modeled as a binomial distribution, is given by

$$P(X \geq 12{,}489) = \sum_{k=12{,}489}^{60{,}000} \binom{60{,}000}{k} \left(\frac{1}{5}\right)^k \left(\frac{4}{5}\right)^{60{,}000-k}.$$

Instead of using the De Moivre-Laplace Limit Theorem (since direct calculation can cause overflow problems), this probability can be calculated using a different method. Take the log of the right hand side, i.e. from,

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k},$$

$$\ln P(X = k) = \ln n! - \ln k! - \ln(n-k)! + k \ln p + (n-k) \ln(1-p).$$

Then use the relationship between the factorial and the gamma function $x! = \Gamma(x+1)$. In Python 3, the math function gamma computes the gamma function.

Example:

```
In [86]: import math

In [87]: math.factorial(5)
Out[87]: 120

In [88]: math.gamma(6)
Out[88]: 120.0
```

Write a Python 3 code pratt_woodruff.py containing the function binpmf (*n*, *p*) to compute the above probability using this method. Plot the probability curve.

4. The Zipf$(n, \alpha = 1)$ random variable X has PMF,

$$P_X(x) = \begin{cases} \dfrac{c(n)}{x} & x = 1, 2, \cdots, n \\ 0 & otherwise \end{cases},$$

where the constant $c(n)$ is set so that $\sum_{x=1}^{n} P_X(x) = 1$. This function is often used to model the popularity of a collection of n objects. For example a Web server can deliver one of n web pages. The pages are numbered such that page 1 is the most requested page, page 2 is the 2nd most requested page, page 3 is the 3rd most requested page, and so on. If page k is requested then $X = k$.

To reduce external network traffic, an ISP gateway caches the k most popular pages. Write a Python function zipfunc (...) to calculate, as a function of n for $1 \leq n \leq 1{,}000$, how large k must be to ensure that the cache can deliver a page with a probability of 0.70, 0.80, and 0.90.

E N D